Towards a New Algorithm for Event Recommendation System

Mahsa Daneshmandmehrabani

Submitted in partial fulfilment
of the requirements for the degree of

Master of Science

Department of Mathematics and Statistics
Brock University
St. Catharines, Ontario

Abstract

We develop a recommendation algorithm for a local entertainment and ticket provider company. The recommender system predicts the score of items, i.e. event, for each user. The special feature of these events, which makes them very different from similar settings, is that they are perishable: each event has a relatively short and specific lifespan. Therefore there is no explicit feedback available for a future event. Moreover, there is a very short description provided for each event and thus the keywords play a more than usual important role in categorizing each event. We provide a hybrid algorithm that utilizes content-based and collaborative filtering recommendations. We also present an axiomatic analysis of our model. These axioms are mostly derived from social choice theory.

Acknowledgements

I owe my deepest gratitude to my supervisor, Dr. Babak Farzad, for his support, patience and broad knowledge. His guidance helped me in my research all the time. I always found him a great source of motivation and encouragement during my studies at Brock University. This work would not have been done without his sincere help and support. He has been an amazing supervisor and a wonderful friend.

I would like to appreciate my thesis committee members: Prof. Henryk Fuks, Prof. Omar Kihel, Prof. Bill Ralph and Prof. Sheridan Houghten for all of their guidance through this process; their discussion, ideas, and feedback have been absolutely invaluable.

I am also thankful to Brock University and the Department of Mathematics and Statistics for their supporting and funding me throughout my studies.

Last but not the least, I would like to thank my family and friends for their endless love, support and encouragement at every stage of my life.

Contents

Chapter 1

Introduction

1.1 Overview

The remarkable growth in the amount of information which is available online can confuse users when trying to find their desired information, services or products on the internet. Therefore, the existence of a system that can help users in a personalized way to find relevant data or items is essential. *Recommender systems* emerged to provide a personalized recommendation for a customer on an online platform to discover her preferred item.

In our daily life, there are many situations in which we should make decisions even without enough data and personal experience. In these cases, we trust in other people's experience and therefore their recommendation. Recommender systems model this natural social behavior for online engines.

A Recommender system can be classified as a subclass of information filtering systems with the goal of guiding users, mostly in a personalized way, to interesting items. Recommender systems have been extensively studied in recent years, and are applied in a variety of applications such as movies, music, news, books, research articles and search queries. A recommender system tries to anticipate the preferences of users in

the most precise way for the items and products they had not yet seen. Since today's world, with the development of information technology, provides people with a spate of information for different alternatives and options, this could be overwhelming and confusing for users lacking an efficient recommender system.

Recommender systems are a personalized information filtering technology which are employed to: predict how much a specific user will like a specific product (prediction problem) or to identify a set of N items that will be of interest to a particular user (top-N recommendation problem).

1.2 Our Contribution

This thesis develops a recommender system for a local ticket provider company, Brüha that recommends events as items to users which are of interest of them. Existing recommendation systems perform well on domain of items such as books, movies, documents and products which can be purchased online. The special characteristics of events which make them different from other kinds of items is that: events are available in the system for a specific time since they have a particular start/end date. Therefore, the events which their end date is passed will be removed from the system. Moreover, each event has a specific location that should be taken into account in computations. Additionally, there is a very short description provided with each event that make the recommendation different from other kinds of document recommendations.

Most of the available recommendation systems utilize two content-based, collaborative filtering or a combination of these two approaches in their computations. We defined a novel approach for our first phase which no information about new users' interests and preferences is available. In this phase, popular events which have higher event's organizer score and also are close in terms of time and location, are recom-

mended to the user.

In the following, we will present symbols and notations in recommendation system literature which are derived from [AT05].

Let $U = \{u_1, \cdots, u_n\}$ be the set of all users and $I = \{i_1, \cdots, i_m\}$ be the set of all items in the system. Note that the sets U and I can have very large sizes. In other words, there can be many users and items available.

A *utility function* evaluates the relevance of item i for user u and returns a ranking score in a totally ordered set R containing nonnegative integers or real numbers. We denote this function by $\hat{r}_{u,i}$, where $\hat{r} : U \times I \to R$.

In a recommendation system, the utility of an item is commonly shown by a rating which reflects how much a user liked a specific item. An application of a recommender system is: for every user $u \in U$, discover an item $i'_u \in I$ such that it maximizes the utility function $\hat{u}$:

$$\forall u \in U, i'_u = \arg\max_{i \in I} \hat{r}_{u,i} \tag{1.1}$$

Typically i'_u is selected from those items which the user has not yet seen.

The utility function can be a proper measure for predicting how much a user will like a particular item. Hence, $\hat{r}_{u,i}$ shows a predicted rating score for item i which user u records.

User u and item i for which the calculation of $\hat{r}_{u,i}$ is done, are called the ***target user*** and the ***target item*** respectively [Ged13].

Definition 1. *The ratings of all users for items can be represented by a matrix $R_{n \times m}$, where $r_{u,i}$ shows the rating score of user u for item i. This matrix is called the* ***rating matrix****, see below figure.*

$$R = \begin{bmatrix} r_{11} & r_{12} & \dots & r_{1n} \\ r_{21} & r_{22} & \dots & r_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ r_{m1} & r_{m2} & \dots & r_{mn} \end{bmatrix} \tag{1.2}$$

Usually, the rating matrix R is very sparse since users record a small number of ratings.

Definition 2. *$\bar{r}_u$ and $\bar{r}_i$ are used to denote the average rating value of user u and item i respectively.*

Remark: There are two different kinds of ratings which are used in recommendation literature: *recorded ratings* and *expected ratings*. Recorded ratings are those ratings which users recorded for seen items and expected ratings are those ratings which a recommender system tries to predict for items that users have not seen.

1.3 Approaches

There are different recommendation systems: *Content-based recommendation*, *Collaborative filtering*, *Demographic recommendation*, *Knowledge-based recommendation* and *Hybrid methods*.

The intuition behind each system is as follows:

- **Content-based recommendation:** This recommender system recommend items to the user similar those ones the user liked in the past.
- **Collaborative filtering:** This recommender system recommends items to the user which users with similar interests and tastes liked before.

- **Demographic recommendation:** This kind of system recommends items based on the demographic profile of the user. The assumption is that different recommendations should be generated for different demographic niches. Many Web sites adopt simple and effective personalization solutions based on demographics. For example, users are referred to particular Web sites based on their language or country. Or suggestions may be customized according to the age of the user. In [WCN12], a demographic recommender system is employed for the recommendation of attractions. This system groups the tourists using their demographic data and then recommends based on demographic classes and their case study is the Trip Advisor website.

- **Knowledge-based recommendation:** This kind of system recommends items based on inferences about a users requirements and preferences. A knowledge-based recommender system recommends products based on particular domain knowledge regarding how a user's needs and preferences can be meet by specific product features. In a knowledge-based recommender system, a similarity function is used to to calculate how much a specific user's needs are close to recommendations.

- **Hybrid methods:** These systems merge two or more recommendation approaches, e.g., content-based and collaborative filtering, to get better results with fewer shortcomings of each approach.

In the following, some of these approaches which are used in this thesis and our model are explained in more details.

1.3.1 Content-based Recommendation

In this system, items are recommended to target user u which are similar to those items that target user u liked before. Most of the content-based recommender sys-

tems are used for recommending items including textual context, such as web sites (URLs) and documents. In a content-based recommendation system, the items which are rated by a user before and their characteristics are studied and a model or profile is created that shows user's interests. Then, the system tries to find and recommend items for which their features are similar to the user's interests and preferences. Typically, in a content-based recommender system, similarity is measured based on items' content. Therefore, a content representation for items is needed to be able to measure this similarity. There are different ways to show the content of an item. For example, if the domain are movies, then each movies content description can be represented by using a list of characteristics such as director, actors, genre, description, and related-titles.

Definition 3. ***ItemProfile(i)*** *or* ***IP(i)*** *is a profile for an item which is a set of features characterizing item i [AT05].*

Usually, ItemProfile(i) is determined by taking a set of attributes out from the item i. Since, content-based approaches are associated with text-based items, the content in ItemProfile(i) are usually keywords. For example, in the Fab system which webpages are recommended, the content-based part of the system represents each web page's content with 100 significant words [BC92].

Therefore, a measure should compute the importance (informativeness) of words within the text content of items to extract keywords of the item's text component which are the best descriptors of the item.

Definition 4. *The* ***importance*** *(informativeness) of word w_i in document d_j is the weighting score $s_{i,j}$ which represents the level of importance in a document.*

One of the best known metrics to determine weighting score which shows how

important a word is to a document in a collection or corpus is *term frequency-inverse document frequency* (*TF-IDF*) measure. This score is a product of two statistics: *term frequency* (TF) and *inverse document frequency* (IDF). The important assumptions regarding this weighting are as follows:

- Words which occur more than once are not less relevant than words which occur once (TF assumption)
- Short documents are not less important than long documents (normalization assumption)
- Words which occur rarely are not less relevant than words which occur frequently (IDF assumption)

Suppose there are D documents that can be suggested to users (documents can be text description of items as well) and also assume that word w_i occurs in m_i of d documents. Additionally, suppose $f_{i,j}$ is the number of appearance of word w_i in the document d_j. Then, $TF_{i,j}$, the term frequency of the word w_i in the document d_j, estimates the importance of word w_i in the document d_j by counting the number of occurrences of word w_i. Moreover, usually a normalization is done by dividing the number of occurrences of word w_i in the document d_j by the number of appearances of the most frequent word in document d_j. $TF_{i,j}$ score is defined as:

$$TF_{i,j} = \frac{f_{i,j}}{\max_z f_{z,j}} \tag{1.3}$$

The intuition behind the inverse document frequency measure $IDF(i)$ is to capture the importance of a word w_i in the whole set of documents. This score reduces the weight of words that appear in many documents such as “the” or “and” since they are usually not proper representatives of documents. Inverse document frequency is usually defined as:

$$IDF_i = \log \frac{D}{m_i} \tag{1.4}$$

Note that D is the total number of documents and m_i is the number of documents containing word w_i.

Then, the *TF-IDF* score which is an importance score for word w_i in the document d_j is obtained by following equation:

$$s_{i,j} = TF_{i,j} \times IDF_i \tag{1.5}$$

Therefore, ItemProfile(i) can be expressed as:

$$\text{ItemProfile}(i) = \vec{\text{IP}} = (s_{1,i}, \cdots, s_{k,i}) \tag{1.6}$$

where k is the number of most important words (keywords) of the document.

As mentioned earlier, content-based methods try to recommend items which are similar to items the user liked previously. Therefore, a study should be done to discover which items are more similar to a user's profile that contains her interests.

Definition 5. ***UserProfile(u)*** *or* ***UP(u)*** *is the the profile of user u including her interests and tastes.*

Profiles of users can be obtained by learning the content and the features of the items already liked by the user. UserProfile(u) is defined as a vector of scores $(s_{1,u}, \cdots, s_{k,u})$, where each component $s_{i,u}$ represents the importance of the keyword w_i to the user u. There are various methods (except TF-IDF) to compute these scores which will be discussed in Chapter 2.

In content-based approach, utility function $\hat{r}_{u,i}$ is specified as:

$$\hat{r}_{u,i} = score(\text{UP(u)}, \text{IP(u)}) \tag{1.7}$$

Both UP(u) and IP(U) can be expressed as *TF-IDF* vectors $\vec{UP}(u)$ and $\vec{IP}(i)$ and a similarity measure, usually cosine similarity, can compute their similarity:

$$\hat{r}_{u,i} = sim(\vec{\text{UP}}(u), \vec{\text{IP}}(i)) = \cos\angle(\vec{\text{UP}}(u), \vec{\text{IP}}(i)) = \frac{\vec{\text{UP}}(u) \cdot \vec{\text{IP}}(i)}{|\vec{\text{UP}}(u)| \times |\vec{\text{IP}}(i)|} \tag{1.8}$$

1.3.2 Collaborative Filtering

As mentioned above, collaborative filtering is a method to filter information which involves collaborations among users. The underlying hypothesis of the collaborative filtering method is that if a person X has the same opinion as a person Y on an issue, it is more likely that X has a similar viewpoint to Y's on a different issue α than to have the same opinion as that of a person chosen randomly.

We clarify this method by giving an example. The following table presents an instance of user-item rating matrix on four movies in 2017. Only a binary "Like/Dislike" rating score is utilized. The goal is to anticipate if user *Bob* will enjoy the movie *Lion* or not.

	Moonlight	Lion	La La Land	Passengers
Bob	Dislike	?	Like	Like
user a	Like	Like	Dislike	Dislike
user b	Dislike		Like	
user c		Dislike	Like	Like
user d	Dislike	Dislike	Like	

According to the above table, users b, c and d have similar opinions to target user *Bob* on common watched movies. In the literature of recommendation systems, similar users are known as *peer users* or *nearest neighbors*. Because both users c and d have the same rating scores on the target movie *Lion*, by using the collaborative

filtering approach, target user *Bob* is expected to dislike it. Then target movie *Lion* is not recommended to the user *Bob*.

Generally, collaborative filtering methods are based on the rating scores of the target user and other users and the main idea is that the predicted rating score of target user x for target item i is probably similar to the rating score of another user y, if x and y have recorded similar rating scores for another item.

There are different methods to implement collaborative filtering recommendation system such as: *Neighborhood-based* and *Model-based*.

- **Neighborhood-based.** In the neighborhood-based (it is introduced as memory-based in [BHK98a] and heuristic-based in [AT05]) collaborative filtering approach, such as Amazon's recommendation engine, user-item rating scores are saved and utilized to anticipate ratings for new items. There are two ways to do this: *User-based* or *Item-based* recommendations.
  - **User-based** systems, such as GroupLens, measure the level of interest of the target user x for a target item i by employing neighbors of the user x, that have analogous rating history on other items.
  - **Item-based** methods, evaluate the rating score of the target user x for the target item i utilizing ratings which the user x recorded for similar items to i. Here similar items means many users have rated those items in an analogous pattern.
- **Model-based.** These kinds of recommendation systems employ rating information to learn a model to make predictions (predictive model). This model can be a data mining or machine learning algorithm. Some of the famous model-based collaborative filtering approaches are: bayesian clustering, support vector machines, and singular value decomposition methods.

Collaborative Recommendation

A common collaborative filtering method's input is the rating matrix R of the system and the output is an estimated rating $\hat{r}_{u,i}$ for an unseen item i from the user u. Users' opinions about the products can be expressed as a rating score usually on a 1-5, 1-7 or 1-10 range which range from "strongly like" to "strongly dislike". Users' ratings can be obtained in two ways: explicitly or implicitly. For example, ratings which users record for products on Amazon.com or MovieLens are explicit ratings and time spent or product reviews are implicit ratings.

A general framework for a collaborative filtering recommendation system is to predict $\hat{r}_{u,i}$ as the following in three main steps:

1. **Neighborhood formation.** The first step is specifying the neighbors of the target user u. As mentioned before, neighbors of a user are those users who had the same opinion regarding other products and items in the past.

2. **Neighborhood selection.** The second step, is selecting k nearest neighbors of the target user u among all neighbors of u. In this step, just neighbors who recorded a rating score for the target item i are considered.

3. **Aggregation of ratings.** In the last step, the k nearest neighbors ratings are taken into account and used to predict rating score $\hat{r}_{u,i}$.

These steps are done for all unseen items for the target user u and then a top-n recommendation list is created putting their rating scores in order. In the following, we study each mentioned step in more detail.

Neighborhood Formation

In this phase, a similarity measure should be employed to discover similar users or items. One of the most common measure to calculate the degree of similarity between

two users x and y is the *Pearson correlation coefficient* and its formula is as the following:

$$sim(x,y) = \frac{\sum_{i \in I}(r_{x,i} - \bar{r_x})(r_{y,i} - \bar{r_y})}{\sqrt{\sum_{i \in I}(r_{x,i} - \bar{r_x})^2}\sqrt{\sum_{i \in I}(r_{y,i} - \bar{r_y})^2}} \tag{1.9}$$

Note that the sum in the above equation only takes items $i \in I$ into account which both users x and y recorded a rating score on them. Since, in practice, the data set is very sparse, a defult rating score, e.g., user's average rating, for items which one of the users has not rated can be used [BHK98b]. In addition to the Pearson correlation coefficient, cosine similarity and adjusted cosine similarity can be used to measure the similarity [JZFF10a].

Cosine similarity is defined as the following:

$$sim(\vec{x},\vec{y}) = \cos \angle(\vec{x},\vec{y}) = \frac{\vec{x} \cdot \vec{y}}{\mid \vec{x} \mid\mid \vec{y} \mid} \tag{1.10}$$

A shortcoming of this measure is that it does not consider the average rating behavior of users. Hence, the adjusted cosine similarity measure is defined:

$$sim(x,y) = \frac{\sum_{u \in U}(r_{u,x} - \bar{r_u})(r_{u,y} - \bar{r_u})}{\sqrt{\sum_{u \in U}(r_{u,x} - \bar{r_u})^2}\sqrt{\sum_{u \in U}(r_{u,y} - \bar{r_u})^2}} \tag{1.11}$$

Neighborhood Selection

One of the most important issues in this stage is the size k which should be selected as the number of nearest neighbors which is not too large or too small.

There are different methods to choose a neighborhood size. Similarity threshold is a technique in which users with a higher similarity are considered.

The authors in [Ged13] propose a user-dependent similarity threshold in which different neighborhood sizes for different users are allowed.

Aggregation of Ratings

In this step, the k-nearest neighbors' ratings are integrated together for the target item i and the rating score $\hat{r}_{u,i}$ is computed based on them. In [RV97] a formula is presented to calculate $\hat{r}_{u,i}$:

$$\hat{r}_{u,i} = \bar{r}_u + \frac{\sum_{n \in N} sim(u,n) * (r_{n,i} - \bar{r}_n)}{\sum_{n \in N} sim(u,n)} \tag{1.12}$$

where N is the set of nearest neighbors of user u.

1.3.3 Hybrid Methods

In hybrid techniques, two or more recommendation methods, e.g., content-based and collaborative, are merged together to avoid each method's limitations.

Different ways are available to mix content-based and collaborative techniques. One way to design a hybrid recommendation system is to apply collaborative and content-based methods separately and combine two outputs from those two systems into the final recommendation by for example a linear combination.

Different hybridization techniques are mentioned in [Bur02]. We explain some of these methods briefly.

Weighted Hybridization

In this technique, the score of the target item is computed based on a combination of its scores which are obtained from each individual recommender in the system. The score for a given item is computed as the weighted sum of scores produced by several recommenders. The most straightforward combination is linear combination of each system's score.

Switching Hybridization

In this approach, the system switches between recommendation systems by using some criterions. For example, in a content/collaborative hybridization, one recommender system is implemented first, i.e. content-based method. If the first employed system, content-based, cannot recommend efficiently, then the other recommender system, collaborative filtering method, is executed. The switching hybrid can avoid problems specific to one method, i.e. the new user problem of content-based recommenders, by switching to a collaborative recommendation system. These kinds of hybridizations add more complexity to the system since switching criteria should be defined. Therefore, it can bring another level of parameterization into the system.

Mixed Hybridization

This method can be used when it is applicable to make a large number of recommendations at the same time. In other words, recommendations from more than one approaches are presented together.

Cascade Hybridization

In this hybridization method, one of the recommendation systems is used first and returns a list of ranked candidate items and then a second recommender system refines the recommendation on the candidate list. For example, consider a restaurant recommender which is a cascaded knowledge-based and collaborative recommender. It uses its knowledge of restaurants to recommends based on the users expressed interests. The recommendations are taken into account with equal preference, and then the collaborative technique is used to refine the recommendation.

1.4 Drawbacks and Shortcomings

In this section, we will discuss about some drawbacks and limitation of the recommendation systems. We study each recommendation system's shortcomings separately.

1.4.1 Content-based Recommendation

Three main drawbacks are:

1. **Limited content analysis.** The content-based recommender is restricted by attributes which are describing items. To have enough features, items content should either be allocated to items manually or be in a format that can be analyzed automatically by a computer (text). Automatic feature and keyword extraction methods perform well on text domains and are hard to apply on for example, graphical images, audio and video streams [AT05]. Another limitation of content analysis is that if two distinct items are expressed by the same attributes then they are not distinguishable and their qualities are not recognizable.

2. **Overspecialization.** These kinds of systems recommend only items whose similarity scores with the user's profile are high and the user will receive only recommendations which are similar to items which have seen or experienced before. Content-based recommender systems cannot recommend an unexpected item to the user [RRS11].

3. **New user.** To create user's profile and recommend efficiently, enough ratings should be gathered to understand the user's interests. For new users which have a little or no rating history, recommended items are not trustworthy [RRS11].

1.4.2 Collaborative Filtering Recommendation

Collaborative filtering techniques have some shortcomings which are propounded in the following:

1. **New users.** This difficulty is similar to the one there is with content-based systems. Since the system should study users' ratings and learns the interests and similarities between users, new users with little or no rating history can make this process hard to apply.

2. **New items.** New items are added to the system over the time and collaborative filtering recommender systems are based on users' interests and their common ratings. A new item can not be recommended until it is rated by a considerable number of users.

3. **Sparsity.** Most of the reommender systems are based on a huge dataset and there are a large number of users and items available. As mentioned previously, the rating matrix M could be very large and also sparsely filled. In particular, the efficiency of a collaborative system depends on a substantial number of users and ratings. For example, in the book recommendation in which there are many books that are rated by just a few users, these books are recommended extremely seldom even when they are highly rated.

Usually, new users and new items problems are called the *cold-start* problem in the recommendation systems literature.

Chapter 2

Background

In this chapter, we provide the general notations and preliminaries for this thesis.

2.1 Similarity Measures

In content-based recommendation systems, we need to measure the similarity between items and users' profile preferences. Moreover, in collaborative filtering methods, a similarity metric is required to measure similarity between the interests of the target user and other users. In this section, we will study different similarity measures and functions which are available.

ItemProfiles and UserProfiles can be represented by vectors in which each component of the vectors represents a corresponding feature of the item. Typically, these components are shown by numeric scores which state that how much the corresponding feature is relevant to that item. Then, the similarity between two vectors $\vec{x}$ and $\vec{y}$ can be measured by the following metrics.

2.1.1 Euclidean Distance

The first and easiest distance measure is the Euclidean distance that calculates the distance between vectors $\vec{x}$ and $\vec{y}$ which is the length of the line segment connecting their tips:

$$d(\vec{x}, \vec{y}) = \sqrt{\sum_{k=1}^{n}(x_k - y_k)^2} \tag{2.1}$$

where n is the number of dimensions (features) and x_k and y_k are the k-th component (feature) of x and y respectively. An extension of Euclidean Distance is the Minkowski Distance which can be obtained by the following formula:

$$d(\vec{x}, \vec{y}) = \Big(\sum_{k=1}^{n}(x_k - y_k)^r\Big)^{\frac{1}{r}} \tag{2.2}$$

In this formula, r is called the degree of the distance. For different values of r, the generic Minkowski distance has a particular name: For $r = 1$, it is called *the city block*, (*Manhattan, taxicab or L1 norm*) distance; For $r = 2$, the distance measure is called *Euclidean distance*; For $r \to \infty$, the measure is called *supremum* (L_{max} *norm* or L_∞ *norm*) distance.

2.1.2 Cosine Similarity Measure

Another technique to calculate the similarity of two vectors is the cosine similarity approach which is a measure of similarity between two vectors of an inner product space that computes the cosine of the angle between them. This method is commonly utilized in the field of text mining and item-based recommendation systems. The similarity between two vectors x and y is defined as follows:

$$sim(\vec{x}, \vec{y}) = \cos\angle(\vec{x}, \vec{y}) = \frac{\vec{x} \cdot \vec{y}}{|\vec{x}| \times |\vec{y}|} \tag{2.3}$$

The value which is computed by this metric is a number between 0 and 1. It

is 0 when two vectors are totally orthogonal and is 1 when both point to the same direction.

2.1.3 Pearson Correlation Coefficient

The Pearson measure is adjusted cosine measure which considers the user's average behavior as well. The similarity between two users a and b by Pearson measure is computed as follows:

$$sim(\vec{x}, \vec{y}) = \frac{\sum_{i \in I}(r_{x,i} - \bar{r_x})(r_{y,i} - \bar{r_y})}{\sqrt{\sum_{i \in I}(r_{x,i} - \bar{r_x})^2}\sqrt{\sum_{i \in I}(r_{y,i} - \bar{r_y})^2}} \tag{2.4}$$

The Pearson correlation coefficient calculates the linear dependence between two variables and returns a value between -1 and 1. Negative values correspond to a negative correlation (low similarity), while positive values indicate a positive correlation (high similarity).

2.1.4 Dice Coefficient

This measure is used to compare the similarity between two text samples a and b and its formula is as follows:

$$sim(a, b) = \frac{2 \times |keywords(a) \cap keywords(b)|}{|keywords(a)| + |keywords(b)|} \tag{2.5}$$

where a and b are the samples which we seek to measure their similarity and the function *keywords* returns keywords of each sample. The possible values for Dice similarity score are between 0 and 1 where the higher value indicates two samples share more common keywords therefore are more similar.

2.1.5 Jaccard Distance

There is another kind of similarity measures which focus on the similarity between sets by looking at the relative size of their intersection. The Jaccard similarity is used to measure how much sets are close. For sets A and B, the Jaccard similarity is defined by $|A \cap B|/|A \cup B|$. Jaccard similarity is denoted by $\text{sim}(A, B)$.

2.2 Different Keyword Extraction Methods

One of the most important problems in the text mining and information retrieval area is to find words and phrases which describe a document well and are significant in the document. These are usually called keywords. There are different methods to extract important words from a given document. We describe some of these techniques in the following.

2.2.1 Chi-squared

This method focus on the keyword extraction algorithm only taking a single document into account rather considering a corpus. In this approach, the first step is to draw the frequent words out. Next, the co-occurrence of each word and frequent words is computed. Two words in the same sentence are called co-occurrence words. A term more probably is a keyword in a document if the probability distribution of their co-occurrence is prejudiced to a specific subset of frequent words. The degree of biases of distribution in this method represents the importance of a word. However, if a word's frequency is small, the degree of biases is not trustworthy. For instance, suppose word a appears only one time and co-occurs only with term x once (probability 1.0). On the other hand, suppose word b appears 100 times and co-occurs only with word x 100 times (with probability 1.0). It is clear that, b sounds more reliably biased. In order to evaluate the statistical significance of biases, a test is employed (This is a common

way for computing biases between expected frequencies and observed frequencies). For every word, the frequency of co-occurrence with the frequent words is considered as a sample value; the null hypothesis is as follows:

"occurrence of frequent terms X is independent from occurrence of term w" that it is expected to reject.

"The unconditional probability of a frequent word $x \in X$ is denoted as the expected probability p_x and the total number of co-occurrences of word w and frequent terms X as n_w. The frequency of co-occurrence of term w and term x is denoted as freq(w, x). The statistical value of χ^2 is defined as:

$$\chi^2(w) = \sum_{x \in X} \frac{(freq(w,x) - n_w p_x)^2}{n_w p_x} \tag{2.6}$$

If $\chi^2(w) > \chi^2_\alpha$, the null hypothesis is rejected with significance level α. The term $n_w p_x$ shows the expected frequency of co-occurrence; and $(\text{freq}(w,x) - n_w p_x)$ shows the difference between observed and expected frequencies. Hence, large $\chi^2(w)$ states that co-occurrence of term w shows strong bias." Generally, words with large χ^2 are important in the document [MI04]. The degree of freedom in this test is $n-1$ where n is the number of classes.

2.2.2 CollabRank

This keyword extraction method takes the keyphrases out by considering common impacts of several documents inside a group context. The first step, *document clustering*, involves categorizing the documents into some clusters utilizing the clustering algorithm. In the second step, *collaborative keyphrase extraction*, for every cluster C, the algorithm extracts keyphrases from each document in two substeps. In the first substep, *cluster-level word evaluation*, a global affinity graph G is built based on all possible terms restricted by syntactic filters in the documents of the given cluster

C, and then the graph-ranking based algorithm is used to evaluate the cluster-level saliency score for every term. Each cluster contains the documents having common and similar subject. In the second substep, *document-level keyphrase extraction*, for every document d in the cluster, evaluate the candidate phrases in the document based on the scores of the words contained in the phrases, and finally choose a few phrases with the highest scores as the keyphrases of the document [WX08].

For a given cluster C, let $G = (V, E)$ be an undirected graph to reflect the relations between words in the cluster. Each vertex in the set of vertices V is a representation for a candidate word in the cluster. E is the set of edges where each edge $e_{i,j}$ in E is associated with an affinity weight $w(v_i, v_j)$ between words v_i and v_j. The weight is computed based on the co-occurrence relation between the two words, controlled by the distance between word occurrences. The co-occurrence relation can relates cohesion relationships between words. Two vertices are connected if the corresponding words co-occur at least once in a window of maximum k words, where $k \in \{2, 3, \ldots, 20\}$. The affinity weight $w(v_i, v_j)$ is simply set to be the count of the controlled co-occurrences between the words v_i and v_j in the whole cluster as follows:

$$w(v_i, v_j) = \sum_{d \in C} count_d(v_i, v_j) \tag{2.7}$$

where $count_d(v_i, v_j)$ is the number of the co-occurrences of words v_i and v_j in document d.

An affinity matrix M is used to describe G where each entry corresponds to the weight of an edge in the graph. $M = (M_{i,j})_{|V| \times |V|}$ is defined as follows:

$$M_{i,j} = \begin{cases} w(v_i, v_j), & \text{if } v_i \text{ is linked with } v_i \text{ and } i \neq j \\ \\ 0, & \text{otherwise} \end{cases}$$

Then M is normalized to $\tilde{M}$ as follows:

$$\tilde{M}_{i,j} = \begin{cases} \frac{M_{i,j}}{\sum_{j=1}^{|V|} M_{i,j}}, & \text{if } \sum_{j=1}^{|V|} M_{i,j} \neq 0 \\ \\ 0, & \text{otherwise} \end{cases}$$

Based on graph G, the cluster-level saliency score $Sclus(v_i)$ for word v_i can be deduced from those of all other words linked with it and it can be formulated in a recursive form as in the PageRank algorithm:

$$Sclus(v_i) = \mu \times \sum_{allj \neq i} Sclus(v_j) \times \tilde{M}_{i,j} + \frac{(1-\mu)}{|V|} \tag{2.8}$$

where μ is the damping factor and usually is considered equal to 0.85, as in the PageRank algorithm.

The matrix form of above formula is as follows:

$$\vec{\kappa} = \mu \tilde{M}^T \vec{\kappa} + \frac{1-\mu}{|V|} \vec{e} \tag{2.9}$$

Where $\vec{\kappa}$ is the words saliency scores vector and $\vec{e}$ is a vector which all elements of it is equal to 1. This problem can be taken into account as a Markov chain in which the words are the states and the corresponding transition matrix is $\vec{\kappa} = \mu \tilde{M}^T \vec{\kappa} + \frac{1-\mu}{|V|} \vec{e}$.

The initial scores of the words are considered as 1.

After the scores for every candidate words in the cluster have been calculated, candidate phrases are selected and evaluated for each single document in the cluster. The score of a candidate phrase p_i is computed by adding the cluster-level saliency scores of the words contained in the phrase.

$$PhraseScore(p_i) = \sum_{v_j \in p_i} Sclus(v_j) \tag{2.10}$$

All the candidate phrases in the document are ranked in decreasing order based on their phrase scores and the top n phrases are selected as the keyphrases of the document.

2.2.3 TF-IDF

In data mining, a famous and basic method to measure the importance of words within a document is TF-IDF approach. This is a value indicating how much a given word is important in a collection of documents. This measure considers two factors to compute the TF-IDF score for each word: *Term frequency and inverse document frequency.* The intuition behind these two factors is as follows. Term frequency states: if a word comes in a document frequently, that word is probably a keyword and has an important meaning in that document. On other hand, Inverse document frequency parameter states that: there are some words such as "the" or "and" which are so common in all documents but do not contribute any significance to documents and cannot be proper candidates for being keywords of a document.

Suppose there is a collection of D documents available. Let $f_{i,j}$ be the frequency of the word i in document j. Then, the term frequency $TF_{i,j}$ is defined as follows:

$$TF_{i,j} = \frac{f_{i,j}}{\max_k f_{k,j}} \tag{2.11}$$

The IDF value for each word is determined as follows.

$$IDF_i = \log_2(D/n_i) \tag{2.12}$$

Where n_i is the number of documents among D documents in which word i appears. Then TF-IDF score is defined as $TF_{i,j} \times IDF_i$. Words which have high TF-IDF score can be selected as keywords of a document.

For example, suppose document d_1 contains words w_1, w_2 and w_3 with frequencies 1,2 and 3 in document d_1 respectively. Moreover, assumne document document d_2 contains words w_1, w_4 and w_5 with frequencies 1,2 and 5 in document d_2 respectively. The TF-IDF score for each word is as follows:

$$TF(1,1) = \frac{1}{3} \tag{2.13}$$

$$IDF(1,1) = \log_2(\frac{2}{2}) = 0 \tag{2.14}$$

$$TF - IDF_{(1,1)} = \frac{1}{3} \times 0 = 0 \tag{2.15}$$

$$TF(2,1) = \frac{2}{3} \tag{2.16}$$

$$IDF(2,1) = \log_2(\frac{2}{1}) = 1 \tag{2.17}$$

$$TF - IDF_{(2,1)} = \frac{2}{3} \times 1 = \frac{2}{3} \tag{2.18}$$

$$TF(3,1) = \frac{3}{3} = 1 \tag{2.19}$$

$$IDF(3,1) = \log_2(\frac{2}{1}) = 1 \tag{2.20}$$

$$TF - IDF_{(3,1)} = 1 \times 1 = 1 \tag{2.21}$$

$$TF(1,2) = \frac{1}{5} \tag{2.22}$$

$$IDF(1,2) = \log_2(\frac{2}{2}) = 0 \tag{2.23}$$

$$TF - IDF_{(1,2)} = \frac{1}{5} \times 0 = 0 \tag{2.24}$$

$$TF(4,2) = \frac{2}{5} \tag{2.25}$$

$$IDF(4,2) = \log_2(\frac{2}{1}) = 1 \tag{2.26}$$

$$TF - IDF_{(4,2)} = \frac{2}{5} \times 1 = \frac{2}{5} \tag{2.27}$$

$$TF(5,2) = \frac{5}{5} = 1 \tag{2.28}$$

$$IDF(5,2) = \log_2(\frac{5}{1}) = 2.32 \tag{2.29}$$

$$TF - IDF_{(5,2)} = 1 \times 2.32 = 2.32 \tag{2.30}$$

2.3 Evaluation of Recommendation Systems

Nowadays, recommendation systems are applicable in many novel applications that expose users to a large number of products. These recommendation systems output a list of recommended products for users, or anticipate how much users might like and prefer each product. Therefore, these systems streamline the process of finding their preferred products.

Every application designer whose goal is to design a recommendation system for an application has a large number of options to choose as her algorithm. Decisions about selecting the most proper approach have been made according to experiments which compare the operation of different recommendation systems. In addition, new

suggested algorithms' performances should be evaluated and compared with available methods. To do so, different evaluation metrics are applied.

There are three different kinds of experiments to evaluate a system:

1. **Offline experiment.** In this type of experiment, which is usually easy to execute, users' behavior is studied and modeled to assess the system's performance. These experiments use available data sets to evaluate. Offline experiments do not need user interaction. Hence they are applied offline with a very low cost. Stored data sets (usually users' preferences) are divided into two sets: a training set and a test set. Then user preferences are allocated to these two sets in a random way. The data in the training set is used to make a user profile and then is used to anticipate user's preference for an item in the test set. In the following, we review various metrics for measuring the prediction quality of a recommender system.

   We describe the most commonly used metrics in offline framework.

   - ***Mean Absolute Error*** (**MAE**): This is a broadly used metric in a recommender system's evaluation process. This metric measures the accuracy of a recommendation approach. It computes the average absolute deviation between a users real (but withheld) rating and the predicted rating by a recommender system. MAE is expressed by the following formula:

     $$\text{MAE} = \frac{\sum_{i=1}^{N} |\hat{r}_i - r_i|}{N} \tag{2.31}$$

     where N is the number of tested item ratings, $\hat{r}_i$ is the rating produced by the recommender system for item i and r_i is the real rating recorded for item i. Lower MAE indicates that a recommender can better predict a user's interest in an item.

- ***Root Mean Squared Error*** (**RMSE**): This is another metric in recommendation systems evaluation literature. This measure calculates the average absolute deviation between the predicted rating by a recommender and a users real (but withheld) rating, but with more emphasis on larger prediction errors. It can be formulated as follow:

$$\text{RMSE} = \sqrt{\frac{\sum_{i=1}^{N} |\hat{r_i} - r_i|^2}{N}} \tag{2.32}$$

  Similar to MAE, N represents the number of predictions, $\hat{r_i}$ is the rating produced by the recommender system for item i and r_i is the real rating recorded for item i. A lower RMSE shows better performance of a recommender system in the prediction process.

- ***Prediction Coverage*** (**COV**): This metric is the percentage of user-item combinations about which a recommender can make predictions. It can be shown as follow:

$$\text{COV} = \sum_{i=1}^{N} \frac{1(\hat{r_i})}{N} \tag{2.33}$$

  where N is the number of tested user-item combinations. The indicator function returns 1 if a prediction can be computed and 0 otherwise.

- ***Precision and Recall:*** The quality of recommender systems which returns a list of top-n recommendations typically is measured by Precision and Recall metrics. To do so, the items set should be divided into two groups: relevant and not relevant. Therefore, in order to decide which item is in which group, ratings should be transformed to a binary scale. A way which is mentioned for this transformation is to consider ratings higher than the user's mean rating as like statements (relevant) and otherwise as dislike statements (not relevant). Then the following notations can be defined to describe Precision and Recall metrics.

- ***ELS:*** The set of existing like statements.
- ***PLS:*** The set of predicted like statements by a recommender system.

$$\text{Precision} = \frac{|PLS \cap ELS|}{PLS} \tag{2.34}$$

$$\text{Recall} = \frac{|PLS \cap ELS|}{ELS} \tag{2.35}$$

The Precision metric shows the probability that an item in the list of recommendations is of interest to the user, whereas the Recall metric indicates how many of the existing like statements were recommended by the recommendation system. In the evaluation process, precision and recall values are computed for all users and then averaged. They should be considered together since improving one is usually at the cost of the other. Hence, the averaged precision and recall measures are combined in the F1-score :

$$\text{F1} = 2 \times \frac{\text{precision} \times \text{recall}}{\text{precision} + \text{recall}} \tag{2.36}$$

2. **User study.** In these types of evaluations, a small number of users are asked to interact with a recommender system, do several tasks and report their experience. While they are doing the tasks, their behavior is recorded. In this process, a number of quantitative measurements such as portion of completed tasks or time spent to complete the tasks is collected.

   A typical instance of such an experiment is to test the effect of a recommendation system on the browsing behavior of news stories. In this example, the people are asked to read a set of stories that are interesting to them such that some of them are related story recommendations and some of them are not in recommendations. We can then check whether the recommendations are used, and whether people read different stories with and without recommendations.

We can collect information such as how many times a recommendation was clicked, and even track eye movement to see whether a person looked at a recommendation. Finally, we can ask qualitative questions such as whether the person thought the recommendations were relevant.

3. **Online experiment.** The last category of evaluation experiments is online experiment. These experiments are large scale experiments on a deployed system and evaluate the performance of the recommenders on real users who are oblivious to the conducted experiment. A recommender system designer hopes to affect the behavior of users. Therefore it is interesting to measure the change in user behavior when interacting with different recommendation systems. For instance, if users of one system follow the recommendations more often, or if the benefit gathered from users of one system exceeds the benefit obtained from users of the other system, then we can infer that one system is better than the other.

   In an online testing system, multiple algorithms can be compared. Such systems redirect a small percentage of the traffic to each different recommendation engine, and study the users interactions with the different systems.

   During running these tests some factors should taken into account: sample (redirect) users in a random way. Moreover, different aspects of the recommender systems should be discriminated. For example, if the algorithm's accuracy is important for us, the user interface should be kept fixed. Or if we want to focus on a better user interface, keep the underlying algorithm should be kept fixed.

   Sometimes, such experiments are risky to run. For instance, a test system that provides irrelevant recommendations can prevent the test users from using the real system again. Hence, the experiment can have a negative influence on the system. Therefore, it is better to run an online evaluation last, after a

vast offline study provides evidence that the candidate systems are reasonable, and perhaps after a user study that measures the users viewpoint towards the system.

Chapter 3

Event Recommender System Design

The goal of developing a recommender system is guiding users to items and products which they may be interested. The general framework to design a recommendation system is as follows: There is a large number of items and users, and also we have information regarding users' feedback about different items. This feedback can be in various types: rating the item, purchasing the item or viewing the item. The purpose is to predict the user's preference for new items which the user has not yet seen and then recommend those products that user probably like.

In this chapter, we will introduce a recommender system model for a ticket provider company, Brüha, in which items are events. A brief description about Brüha is found in Appendix B. One of the special features of events which make them different from other kinds of items and products is that they are perishable items and it means they have a relatively short and specific lifespan. Therefore, there is no explicit feedback about events available. Moreover, there are two particular and considerable features of events that we should take into account in this setting: *time* and *location*. Every event has a specific beginning time and location. We develop a

hybrid-recommender system which is a weighted combination of different approaches: *content-based recommendation* and *collaborative filtering*. This hybrid recommender system itself is a combination of two hybridization approaches, weighted and switching. Our algorithm is a location-aware recommender system which tracks users' latitude and longitude. Our proposed system follows users and gathers their location data and creates a profile of interests according to events which they have attended.

We define four different phases for our work based on the amount of data we have about users' interests and preferences:

1. **LTPO (location, time, popularity and organize score) recommendation (Cold start phase):** This phase is implemented for new users about whom we do not know the interests and preferences.

2. **Content-based recommendation:** This step is executed when we have data about users' activities such as purchasing of tickets for some events that indicate her interest.

3. **Collaborative filtering:** This phase of the algorithm is applied after a while when more information about users and their interactions with items and also other users is available.

4. **WS (Weighted-Switching) Hybrid recommendation:** In this stage, the content-based and collaborative filtering information are merged and the recommendation process will be done based on items' hybrid scores.

In the next sections, we will study each phase in more detail.

3.1 Phase 1: LTPO recommendation

In this phase, we study the case that users are new to the system and there is no information available about their interests and preferences. In other words, there

is no history available about their online activities and ticket purchases on their Brüha's profiles. Therefore, existing approaches such as collaborative and content-based filtering to design a suitable recommendation systems which focus on the history of user's interaction and preferences fail in this phase. To tackle the cold start problem in this phase, our model recommends events which are popular and have a higher *organizer's score* (*quality score*) and also are close to the user in terms of time and location. There are some factors that can indicate how much an event is popular or how an organizer's score can be obtained which will be discussed in the following.

We design a new model, *LTPO recommendation (location, time, popularity and organize score aware recommeder)* that recommend events to users according to four major factors:

- Event's time
- Event's location
- Popularity of event e_i
- The score of the event's organizer

We will assign a score to each event and recommend events chosen randomly with probabilities proportional to their scores to a user who is searching for events to attend. We formulize our model as following:

$$score(e_i) = f_{e_i} \times p_{e_i}^{\alpha} \times q_{e_i}^{\beta} \times a^{\lambda} \times b^{\delta} \tag{3.1}$$

where :

- **Feasibility score** f_{e_i}: This factor is the feasibility factor of the event e_i and is equal to 0 if the time that the user is searching an event for that, is after the end date of the event e_i or whenever the location of the event e_i is far away

from the user but the time for this event is for example tonight and is equal to 1 otherwise.

- **Popularity score** p_{e_i}: This is the popularity score of the event e_i. We obtain this score by considering the capacity of the event's location. In other words, $p_{e_i} = log(c_{e_i})$, where c_{e_i} is the capacity of the event's location. This score will be in the $(1-5)$ range. The default value for this score is 3.

- **Quality score** q_{e_i}: This is the quality score assigned by Brüha to the organizer of event e_i. This score is based on different factors such as the number of participants in the last events of that organizer and the revenue generated by it. We suggest Brüha to use a $0-6$ scale for this assignment. The default values for this score is 2 or 3. Score 2 is for those who have not posted any event so far (cold start problem) and score 3 is for those who have a history. Score 0 is for dead posts, score 1 shows a poor event advertisement, score 4, 5 and 6 represents good, very good and excellent advertisements respectively.

- a: This parameter represents the distance between the user's and the event's locations. We assign a number in the $[1, 18]$ range to this factor according to distance difference. To do this, we define 18 different ranges as following:

$$a = \begin{cases} 18, & 0\text{km} < d \leq 1\text{km} \\ 17, & 1\text{km} < d \leq 2\text{km} \\ \vdots & \\ 1, & d > 60\text{km} \end{cases}$$

where d is distance difference (in kilometers) between user's and event's location.

- b: This factor represents the time difference between the user's desired time and event's start date. We assign a number in the $[1, 11]$ range to this factor according to time difference. To do this, we define 11 different ranges as following:

$$b = \begin{cases} 11, & 0\text{s} < t \leq 86400\text{s} \\ 10, & 86400\text{s} < t \leq 86400\text{s} \times 2 = 172800s \\ \vdots & \\ 1, & t > 86400s \times 28 = 50803200\text{s} \end{cases}$$

  where t is the time difference (in seconds) between the time of the event and the time that users search an event for that. We supposed that all times are in unix time stamp format.

  Complete tables for parameters a and b are given in appendix C. To obtain parameters a and b which indicate the difference between the location and the time of the user u and event e respectively, we picked 10 users as samples and computed the Euclidean distance and time difference between their locations and time to see how many intervals should be defined and what score should be assigned to each event.

- $\alpha, \beta, \lambda, \delta$: These are the parameters that we use as powers for each factor to tune the model over the course of time. Based on our experiments, $\alpha = 1, \beta = 2, \lambda = 1, \delta = 5$ result reasonable scores for a sample of 10 users and 136 events. We believe that distance factor is more important than time factor to users. Hence the value for distance factor's exponent is larger than time factor's exponent. Moreover, Brüha has some preferred organizers that like their events are shown

to users with high probabilities. Therefore, the exponent for quality score is higher that exponent of popularity score.

3.2 Phase 2: Content-based Recommendation

As mentioned in the first chapter, there are two main approaches for building a recommendation system.

- *Content-Based* recommendation systems are based on finding similarity between features of items and the users' profiles.
- *Collaborative- Filtering* recommendation system focus on similarity between users' interests.

In the content-based filtering technique, descriptions of items (events and venues) and users' profiles (containing information about their preferences) are studied. In this method, keywords are utilized to describe items and also a profile is created for each user to record what type of item she is interested in. Then, a content-based recommender system tries to find items which are more similar to the user's taste by measuring similarity between a new item's attributes and those ones the user liked in the past. In other words, different events are compared with events which were selected by the user previously and then the most similar ones are recommended to the user.

In this phase, we study the case that information about preferences and interests of users are available (explicitly or implicitly) and we can compare the event's features with the user profile and recommend events which are similar to the user's taste. To do this, we need to measure similarity between features of events and users' interests. There are different approaches to measure the similarity between events description and user's profile which are explained in Chapter 2. We choose cosine similarity

method (because this approach is simple to apply and average rating which is used in Pearson measure is not defined in this phase) to compute similarity between user's profile and event's features.

To be able to use the cosine similarity we should define vectors that describe the user's interests and event's features. We define these vectors as follows:

Let $\vec{v}_e$ and $\vec{v}_u$ be the event's and user's *descriptor vector* respectively. Each element of these vectors represents one of the categories that can be defined on events descriptions. We have defined 13 different categories for events (i.e. film, art, drink/food, sport/fitness, etc). The defined table of categories and keywords are given in Appendix D. The values for these elements are each category's score for that event. We will discuss how these scores can be obtained in the following.

3.2.1 Score of Categories

To compute each category's score, we need to have keywords which are usually associated with these categories. For example, for a given event in drink/food category some possible keywords which can appear in the description of that event are: beer, brewery, bar, pub, cafe, cheese, pizza, chef, etc. We studied 136 sample events and obtained each event's keywords and concepts. Then, we studied all these keywords and concepts and extracted some important keywords which are more likely to appear in descriptions and created a file containing all 13 categories and their associated keywords. This file can be updated over the time with having more events. Now we can compute an event's descriptor vector and measure its similarity with a user's descriptor vector.

Events' Descriptions Challenges

In our work, we are dealing with short documents (events' descriptions) which may contain less than 200 words. The most important challenge of our research is due to

the short documents' feature: terms appear in the document only once. Therefore, previous methods which rely mostly on word frequency cannot give accurate result and hence become less efficient. Moreover, collabRank and chi-squared approaches are depend on term co-occurrence and term frequencies which are two factors of long documents and text files. For these reasons, we use the idea of the model which is proposed in [TTK+12].

3.2.2 Extracting Keywords and Their Scores Inspiring IKE Algorithm

To be able to compare user's and event's descriptor vectors, we should extract important terms from event description and assign an importance score to them. To do this, we apply the *IKE (Informativeness-based Keyword Extraction)* approach which evaluates word informativeness in various abstraction degrees. The algorithm has two main steps:

1. **Preprocessing and categorizing.** In this step, text descriptions (events' descriptions) will be preprocessed by removing stop words and words with less than 3 characters. Then we categorize the text descriptions by using agglomerative (CompleteLink) categorizing in which every text description establishes its own cluster or group and in every step, the most analogous descriptions will be added to the category till at most s categories are available [TTK+12].

2. **Word informativeness computation.** This process will study words which are in two distinct levels: words that are more abstract but more commonly appear in description such as "Rock & Roll", and ones that are more expressive but more infrequent such as "Rambo". This evaluation is divided into three levels: *corpus level*, *category level* and *description level*.

   Input: the set of all events' descriptions(corpus) and keywords

Output: Keywords' importance scores

- **Corpus level evaluation**. In this level, the goal is to find words that describe the event's description in a more abstract level. These kinds of words, usually appear commonly (but not too common). For example, we aim to discover words like "Rock & Roll" rather than only "music". Here an imporatant assumption is: the most significant words in this level are words that are neither too frequent nor too scarce in the corpus. Authors in [TTK+12] use this idea to find these kinds of words: detecting words that their IDF (inverse document frequency) score is close to the supposed optimal value for words that are not too frequent nor too rare.

  Then the corpus level score $s_{corpus}(w)$ for word w is computed by using following formula:

$$s_{corpus}(w) = IDF(w) - IDF_{opt}(w) \tag{3.4}$$

  where $IDF_{opt}(w)$ is the supposed optimal IDF which is calculated by the following equation:

$$IDF_{opt}(w) = \mu \times \left| \log \frac{wf_c}{n_0} \right| \tag{3.5}$$

  where wf_c is the word's frequency in corpus and $n_0 = 0.03 \times |D|$ which states that a word is optimally informative in the corpus level when it appears in 3% of descriptions. D is the set of all descriptions. Also, parameter μ is used to accent terms are more scarce than common. In [TTK+12] the authors choose $\mu = 1.1$.

- **Category level evaluation.** In this level, the importance of a word is studied in category level. By using the authors' idea in [TTK+12] we

categorize the descriptions that have the same topic and then find terms which are important in the category. If the word w appears many times in the category c but rarely in other categories, it can be said that the word is significant in category c. To compute this score for each keyword, firstly, the text descriptions are broken into shorter parts called *fragments*. These description fragments are identified using breaks such as question marks, commas, semicolons, other similar characters and words such as "and" or "or".

The category level score has two parts: SCoR and SCaT; relevance score of the word in the corpus, and in the category, respectively.

- **SCoR.** This part of the score is calculated by the following equation:

$$SCoR = IAFL(w) + IC(w) \tag{3.6}$$

where $IAFL(w)$ is the inverse average fragment length for word w and is obtained by

$$IAFL(w) = \frac{1}{\frac{1}{|f_w|}\sum l_f(w)} \tag{3.7}$$

where l_f is the length of the fragment f, and $|f_w|$ is the number of fragments in which word w appears in them.

$IC(w)$ is the inverse category count and is calculated by

$$IC(w) = \frac{1}{c_w} \tag{3.8}$$

where c_w is the number of different categories in which word w occurs.

- **SCaT.** This score represents the informativeness of the word in categories. As the authors in [TTK+12] mentioned, the purpose is to detect words that occur often within the category and seldom in other

categories. This measure consists of two probabilities: $P(d|c)$, the probability for the document d containing word w within the category c and $P(c|d)$, the probability that document d with word w occurs in category c. In the other words, these probabilities can be obtained as follows:

$$P(c|d) = \frac{|\{d : w \in d, d \in c\}|}{\{d : w \in d\}} \tag{3.9}$$

and

$$P(d|c) = \frac{|\{d : w \in d, d \in c\}|}{\{d : d \in c\}} \tag{3.10}$$

Then $SCaT$ for the word w in the category c is computed as follows:

$$SCaT(w) = P(c|d) + P(d|c) \tag{3.11}$$

And Finally, the category level score is obtained by

$$s_{category}(w, c) = SCaT \times SCoR \tag{3.12}$$

- **Description level evaluation.** To find the most important words in a description, by using the corpus and category level scores its importance score is calculated. To be able to compare corpus level score with category level scores, corpus scores are normalized to fall in the [0,1] range. Finally, the document level score for word t in document d calculated by following equation:

$$s_{n,corpus}(w) = \frac{s_{corpus}(w)}{\max_{\text{all words}} s_{corpus}(w)} \tag{3.13}$$

$$s_{doc}(w,d) = \eta \times s_{category}(w,c) + (1-\eta)s_{n,corpus} \tag{3.14}$$

where η is the weight that shows giving more weight to either category or corpus.

3.2.3 Descriptor Vectors of Events

Once we have extracted keywords of a given event's description and their importance scores, we should determine the values for the descriptor vector of the event.

Let $\vec{v_e} = (e_1, e_2, \cdots, e_s)$, where s is the number of categories, be the descriptor vector for the event e. Suppose for the given event e, the set of keywords K with their importance scores are extracted by the IKE approach. We want to assign a value to each category (each element of the vector) according to the importance scores of the keywords. At first, we check which keyword belongs to which category. Suppose at the end, for each category c_i we have the set of keywords K_i where $\cup K_i = K$. The value of each c_i denoted by e'_i can be defined as follows:

$$e'_i = \sum_{k_i \in K_i} score(k_i) \tag{3.15}$$

Then each element of the descriptor vector $\vec{v_e} = (e_1, \ldots, e_s)$ is defined as:

$$e_i = \frac{\sum_{k_i \in K_i} score(k_i)}{\sum_{e'_i} \sum_{k_i \in K_i} score(k_i)} \tag{3.16}$$

3.2.4 Descriptor Vector of Users

In this section we introduce a method to obtain users' descriptor vector $\vec{v}_u$. Recall that there are s categories and let $\vec{v}_u = (u_1, \ldots, u_s)$ be the descriptor vector of user u where each u_i represents the score of the i-th category for user u. In the following

we explain how we calculate each score u_i.

Users can select their favorite categories by clicking on category icons in their profile. We utilize *binomial distribution* to model the selection of categories to figure out how many categories should be selected by a user so that we can be assured her selection is comprehensive. We note that it is very likely that users do not thoroughly consider all categories. That is in many cases, their selection is only a subset of their true interests. Moreover, we are conscious that an important assumption in binomial distribution is the independence of experiments. Unfortunately in our setting, selecting categories may not be independent trials. For example, if a person selects *music* category as one of her interests, it is more likely that she selects the *party* category as well. However, the binomial distribution is applied to model the user's selection. In other words, we want to find the threshold for the number of selected categories whose selection provide broad information regarding her interests.

The binomial distribution with parameters n and p returns the discrete probability distribution of getting exactly k successes out of n independent Bernoulli experiments (i.e. success/failure experiments) where the probability of success in each experiment is equal to p and the probability of failure is $1 - p$. For the random variable X which follows the binomial distribution with parameters n and $p \in [0, 1]$, the probability of getting at most k successes in n independent experiments is called the *cumulative distribution function* denoted by *CDF* and is computed as follows:

$$P(X \leq k) = \sum_{i=0}^{k} \binom{n}{i} p^i (1-p)^{n-i} \tag{3.17}$$

For example, if there are 20 event categories and the user select just 3 categories as her interests, this does not necessarily mean that the user is not interested in another 17 categories and will not attend in events which fall in those 17 categories. However, if a user selects 12 categories among 20 categories, we can say that the user is not interested in the 8 remaining categories with very high probability.

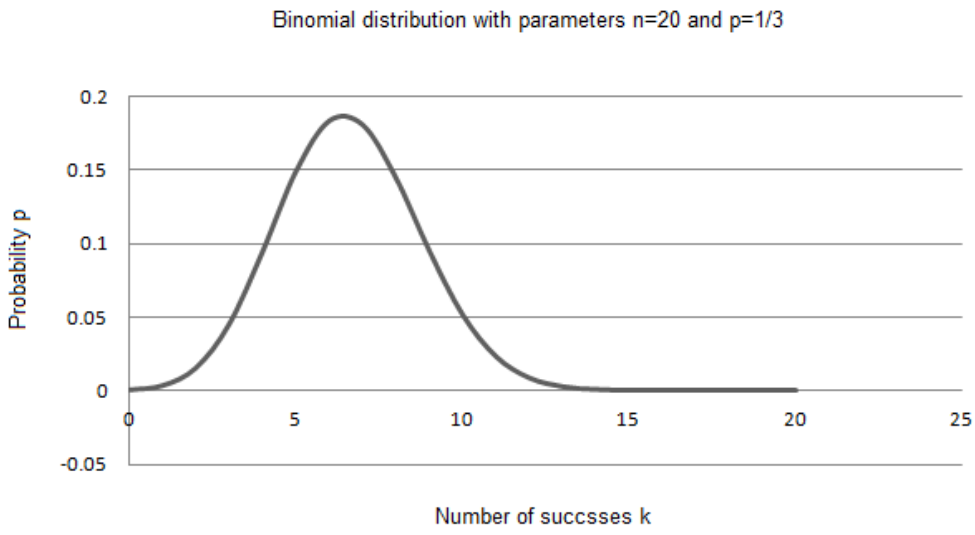

Figure 3.1: Binomial distribution of users' behavior

We observed that on average, users are interested in $\frac{1}{3}$ of the total categories. Therefore, $p = \frac{1}{3}$ and n is the total number of categories. The figure in the following is the binomial distribution with parameters $n = 20$ and $p = \frac{1}{3}$ which is applied to compute the number of successes $k \in \{0, 1, 2, \ldots, 20\}$.

According to the results which are drawn from the above Figure, the cumulative distribution function for $k = 10$ is about 0.96. This means the probability of getting $k \geq 10$ is less than 5%. In our setting, it means that if a user selects at least 50% of the categories, we can say that she is not interested in the rest of the categories with a high probability. We suggest the threshold $t = 50\%$ as a user's number of selections which includes comprehensive information about her tastes.

Let S be the set of categories selected by the user. We consider two cases to compute each score u_i.

1. **Case 1**: If $|S| \geq t \times s$:

   In this case, we can be confident that the user's selection size is large enough and therefore is comprehensive. Hence, we can assign the 0 score to the unselected

categories and $\frac{1}{|S|}$ score to each selected category.

$$u_i = \begin{cases} \frac{1}{|S|}, & \text{if } i \in S \\ \\ 0, & \text{if } i \notin S \end{cases}$$

2. **Case 2**: If $|S| < t \times s$:

   In this case, the size of the user's selection is not large enough. Therefore, unselected categories are not necessarily those ones that the user does not like. Thus we consider a chance (proportional to their scores) for unselected categories to be shown in the recommended list.

$$u_i = \begin{cases} \frac{t \times s + |S|}{2 \times t \times |S| \times s}, & \text{if } i \in S \\ \\ \frac{1 - \frac{t \times s + |S|}{2 \times t \times |S| \times s}}{s - |S|}, & \text{if } i \notin S \end{cases}$$

In addition to selecting categories, on Brüha's platform, users can enter some keywords which are important to them and can describe their interests and preferences better. We study these cases in the following.

Providing Keywords by Users

Users can write some keywords which describe their tastes. These keywords may belong to categories selected by that user and exist in our table. Suppose for each

category c_i, the user has entered k_i keywords in her profile. We believe that providing keywords will increase the score of the corresponding category. Therefore, the score which is assigned to each category is not equally distributed. This score is proportional to the weight representing emphasis of that category according to the number of keywords which user has provided. Since entering keywords is a time consuming action, we assume that it indicates a stronger interest than that by clicking on a category. We suggest the weight $(1 + 2 \times k_i)$ for each category c_i. Then we have:

1. **Case 1**: If $|S| \geq t \times s$:

$$u_i = \begin{cases} \dfrac{1}{\sum_{i=1}^{s}(1 + 2 \times k_i)} \times (1 + 2 \times k_i), & \text{if } i \in S \\ 0, & \text{if } i \notin S \end{cases}$$

2. **Case 2**: If $|S| < t \times s$:

$$u_i = \begin{cases} \dfrac{\dfrac{t \times s + |S|}{2 \times t \times s}}{\sum_{i=1}^{s}(1 + 2 \times k_i)} \times (1 + 2 \times k_i), & \text{if } i \in S \\ \dfrac{1 - \dfrac{\dfrac{t \times s + |S|}{2 \times t \times s}}{\sum_{i=1}^{s}(1 + 2 \times k_i)} \times (1 + 2 \times k_i)}{s - |S|}, & \text{if } i \notin S \end{cases}$$

Remark: Written keywords may belong to categories selected by the user but do

not exist in our table. In this case, we ignore the keywords at the moment but then add the keywords manually to our table if they are meaningful. In addition, keywords may belong to unselected categories. In this case we can assume the corresponding category is selected by the user.

3.2.5 Similarity of Events' and Users' Descriptor Vectors

Now we can measure the similarity between the user's vector and the events' vectors by the cosine approach and recommend events with higher similarity to users. To do so, the cosine similarity metric is employed.

$$sim(\vec{v_u}, \vec{v_u}) = \frac{\vec{v_u} \cdot \vec{v_u}}{|\vec{v_u}| \times |\vec{v_e}|} \tag{3.18}$$

Then the content-based score of the event e for target user u is defined as follows:

$$S_{u,e,CF} = sim(\vec{v_u}, \vec{v_e}) \tag{3.19}$$

3.3 Phase 3: Collaborative Filtering

As it is mentioned earlier, collaborative recommendation systems use information about users' behavior or ratings in the past. Then this information is employed to predict a specific user's rating for a particular item.

In collaborative systems, a rating matrix of existing users and items is studied as input. Then the system returns a prediction score for a certain item from a specific user.

We use the idea of one of the earliest approaches which is called *user-based nearest neighbor recommendation*. The reader is referred to [JZFF10b] for additional information.

A collaborative system gets data regarding the degree of users' interests and users' ID as input. Additionally, the system discover peer users (users with similar interests to the target user). Then, for every event e which the target user u has not decided to attend, a prediction is calculated based on if the peer users have shown interest or purchased tickets for the event e or not. There are two basic assumptions in this approach:

1. If users' tastes were similar before, they will be similar in the future as well.

2. Users' interests and preferences stay stable and consistent.

Remark: There is one important feature of events which makes our approach different from a typical collaborative recommendation system. Unlike other products in other applications such as books or movies in which the other users' ratings and opinions about them may be available, in Brüha's setting peer users' opinions about new posted and upcoming events are not available for a while. Our proposed solution to cope with this problem is defining parameter γ in the hybridization process (it will be discussed in section 3.4) which considers the reactions to upcoming events.

Let $U = \{u_1, \ldots, u_n\}$ and $E = \{e_1, \ldots, e_m\}$ be the set of all users and events in the system, respectively. Note that there are two kinds of events: events which have tickets and those ones that do not have tickets. Moreover, let $M_{n\times m}$ be the a matrix which stores users interests information. Each element $m_{i,j}$ is obtained as follows:

1. **Case 1**: If event e_j does not have tickets:

$$m_{i,j} = \begin{cases} 1, & \text{if user } u_i \text{ has liked event } e_j \\ 0, & \text{if user } u_i \text{ has not liked event } e_j \end{cases}$$

2. **Case 2:** If event e_j has tickets:

$$m_{i,j} = \begin{cases} 2, & \text{if user } u_i \text{ has bought the ticket of event } e_j \\ 1, & \text{if user } u_i \text{ has liked event } e_j \\ 0, & \text{if user } u_i \text{ has not liked event } e_j \end{cases}$$

To make these two kinds of ratings comparable with each other, we normalize all of the $m_{i,j}$'s. To do so, we divide all $m_{i,j}$'s by the maximum value.

$$m'_{i,j} \leftarrow \frac{m_{i,j}}{\max_j m_{i,j}} \tag{3.20}$$

If a particular user u_i has not decided to attend event e_j, the corresponding element $m_{i,j}$ stays empty. Note that matrix M is defined based on rating matrix R which is introduced in chapter 1.

Remark: If we could store the period of time which each user spend to check events and read their descriptions, it could be very helpful to measure the degree of users' interests in events.

Pearson correlation coefficient, one of the most common similarity measure, is used to find peer users with similar tastes. The similarity between the target user u and user $x \in U \setminus \{u\}$ denoted by $sim(u, x)$ is computed by the following formula:

$$sim(u,x) = \frac{\sum_{e \in E}(m'_{u,e} - \bar{m'_u})(m'_{x,e} - \bar{m'_x})}{\sqrt{\sum_{e \in E}(m'_{u,e} - \bar{m'_u})^2}\sqrt{\sum_{e \in E}(m'_{x,e} - \bar{m'_x})^2}} \tag{3.21}$$

In the above formula, $\bar{m_u}$ is the average rating of user u. Peer users are those ones whose similarity scores with the target user are higher than a specific threshold.

To predict if user u is interested in event e and would like to attend at it, the following formula is employed:

$$\hat{m}_{u,e} = \bar{m'}_u + \frac{\sum_{a \in N} sim(u,a) * (m'_{a,e} - \bar{m'}_a)}{\sum_{a \in N} sim(u,a)} \tag{3.22}$$

where N is the set of nearest neighbors of user u. These N nearest neighbors are first N users with the highest similarity score with target user u. We suggest $N = 15$ for our work. Therefore, $\hat{m}_{u,e}$ is the predicted score which is obtained from the collaborative filtering method. We refer to this score as $S_{u,e,CF}$. In other words, this score describes how much the target user u will like event e by employing the collaborative filtering approach and is equal to $\hat{m}_{u,e}$ which is computed above.

$$S_{u,e,CF} = \hat{m}_{u,e} \tag{3.23}$$

3.4 Phase 4: WS Hybrid Recommendation

A hybrid recommendation system incorporates two systems to increase the overall performance. The main reason of using and combining multiple recommendation methods is taking advantage of and suppressing the drawbacks of each individual system.

Remark: Note that the hybridization process is meaningful if we could implement the collaborative filtering approach. In other words, we apply the hybrid recommendation whenever ratings (explicit or implicit) for recently posted events are available otherwise the system's recommendation would be based on the content-based recom-

mender's score.

Our proposed hybridization approach, *WS hybrid recommendation*, is a combination of the weighted and the switching hybridization methods which were discussed in Chapter 2. The weighted approach merge two content-based and collaborative filtering scores and switching approach is applied to switch between content-based and collaborative filtering systems whenever ratings (explicit or implicit) with new posted events are not available. The goal of a WS hybrid recommender system is predicting target event e's score for target user u by using information derived from content-based and collaborative components. Let $S_{u,e,CB}$ and $S_{u,e,CF}$ be the calculated score of target event e for target user u in content-based and collaborative phases, respectively. Then, the hybrid score $S_{u,e,H}$ of target event e for target user u would be a linear combination of scores $S_{u,e,CB}$ and $S_{u,e,CF}$ with hybridization parameter γ. This parameter controls the contributions of each component and switching criteria. Score $S_{u,e,H}$ can be obtained by the following equation.

$$S_{u,e,H} = \gamma \times S_{u,e,CB} + (1 - \gamma) \times S_{u,e,CF} \tag{3.24}$$

According to this equation, the higher value of γ indicates the greater weight and impact of the content-based component on the hybrid score. For example, if $\gamma = 1$, it means the hybrid score depends only on the content-based score or $\gamma = 0$ shows the dependence of the hybrid score only on the collaborative-filtering component of the system. To define the switching criteria which is based on number of ratings for the event, define:

$$\gamma = \frac{1}{1 + \ln(\text{number of explicit or implicit ratings on event e} + 1)} \tag{3.25}$$

If the number of ratings for target event e increases, γ decreases and therefore the

collaborative filtering component's score increases.

Chapter 4

Axiomatic Analysis

In this chapter, we do an axiomatic analysis on our recommendation model and introduce some properties which are satisfied by our model. Most of these properties are the results derived from Social Choice theory. Social Choice theory is a theoretical setting which deals with aggregation of individual opinions and preferences into a collective and social decision. An aggregation function f seeks to combine preferences or utilities instead of ratings.

In the following, we will study some properties which are satisfied by our model and mostly are based on social choice theory. These arguments can be useful in group decision making when the goal of a system is recommending events to a group of users.

Definition 6. ***Hybrid-score matrix*** *$H_{n\times m}$ is a matrix where $h_{i,j}$ represents the hybrid score of event e_j for user u_i.*

The first property is derived from the weak Pareto property which states that if all members of a society prefer alternative x over y, then it can be inferred that x is socially chosen over y. We claim the collaborative component of our model satisfies this property. It means that if all users except target user u prefer event e over event

e', then it implies that target user u will also prefer event e over event e'.

Property 1. *For all users $v \in U$ who are distinct from target user u and two specific events e and e', if $m'_{v,e} > m'_{v,e'}$ then $S_{u,e,CF} > S_{u,e',CF}$.*

If all users except target user u, prefer event e over event e', then nearest neighbors (the most similar users to target user u) of the target user also prefer event e over event e'. Then, according to the collaborative filtering computations target user u prefers event e over event e' as well.

This property is not satisfied by the content-based component of our model. The reason is that although all users except target user u prefer event e over event e', event e''s features may be more similar to those which the target user u likes. Therefore, it can not be concluded that target user u prefers event e over event e'.

The next property is derived from the *independence of irrelevant alternatives* condition in social choice theory.

Property 2. *If target user u prefers event a to event b which means $S_{u,a,CF} > S_{u,b,CF}$, then if a new event c is added to the system, the preference of event a over event b may change.*

In [PHG+00], a similar property is mentioned that in our setting can be described as follows:

If ratings on all events which target user u has rated ($m_{u,e}$ for all events e for which u has bought a ticket or liked) and also for two specific events a and b which target user u has not rated, remain unchanged, then any change in other ratings will not change the relative ranking between two events a and b. This property is true

because the ratings for events which target user u has rated show how similar target user u is to other users. In other words, ratings for events which target user u has not rated do not affect this similarity measure (here rating means buying a ticket of the event or liking the event). Therefore, the predicted relative ranking between two events a and b only depend on the ratings for events a and b. For example, if target user u has not rated event e, then other users' opinions about event e have no impact on the ranking between two events a and b. However, a critical thinking about this property is that: if a new event c is added to the system, the target user's rating for event c will change the similarity score of user u and other users and as a result the relative ranking between events a and b may change. For example, suppose there are three users u, j and k and seven events in the system such that their ratings are as follows:

	a	b	d	e	f	g	h
user u	?	?	4	1	1	2	1
user j	3	1	3	2	1	2	2
user k	1	3	1	5	1	4	1

Note that the ratings of user u for event a and b are not recorded and we want to predict whether event a is preferred over b or vice versa. For simplicity, we consider 1-5 rating scale. Before event c is added, the similarity scores are $sim(u, j) = 0.8135$ and $sim(u, k) = 0.3344$. It means that user u prefers event a over event b (similar to user j). Suppose event c is added to the system and the ratings for that are as follows:

	a	b	c	d	e	f	g	h
user u	?	?	5	4	1	1	2	1
user j	3	1	1	3	2	1	2	2
user k	1	3	5	1	5	1	4	1

After adding event c to the system, $sim(u, j) = -0.5563$ and $sim(u, k) = 0.5782$. It means that user u prefers event b over event a (similar to user k).

The next property is inspired from the positive response condition in social choice theory. This condition states that an ordering of a society on different alternatives responds positively to changes in each member's value of alternatives. Therefore, if one alternative's value rises or remains still in the ordering of every individual without any other change in those orderings, it is expected that it rises, or at least does not fall, in the social ordering.

Property 3. *Suppose H and H' are two hybrid-score matrices, such that for some $u_i \in U$ and $e_k, e_l \in E$, $h_{i,k} > h_{i,l}$ and $h'_{i,k} > h_{i,k}$ and for all $u_j \in U$ where $u_j \neq u_i$ and all $m \in E$, $h'_{j,m} = h_{j,m}$. Then $h'_{i,k} > h'_{i,l}$.*

Chapter 5

Conclusion and Further Research

In this thesis, we designed an event recommender system which is a new model in terms of items' special features. Events have particular characteristics which make them different from other kinds of items such as books or movies. They have a specific lifespan and are not always available in the system. Furthermore, a short text (event's description) is associated with each event which makes the keyword extraction task different from other keyword extraction methods. Moreover, every event has a specified time and location. Our model contains four main phases: cold-start phase, content- based recommendation, collaborative filtering and WS hybrid recommendation. In the cold-start phase, when the user is new and has no activity in the system, events are recommended with probability proportional to their score based on four factors: time, location, event popularity score and organizer score. In the content-based phase, the events are recommended which are more similar to user's interests and tastes. In this thesis, we defined event and user descriptor vectors and methods to obtain the values (scores) of these vectors' components. Then their similarity is measured by cosine similarity measure and this similarity defines the content-based recommendation score of the event and the user. In the collaborative filtering phase, the most similar users' opinions are used to predict if the target user

is interested in the target event. To do this, events are divided in two groups: with tickets and without tickets. Then for each group a rating setting is defined to be able to use the collaborative filtering approach. Finally, in the WS hybrid recommendation phase, a new hybridization approach is employed which is a combination of two weighted and switching approaches with hybridization and switching criteria γ which is defined in Chapter 3. Then, in Chapter 4, a set of properties which are satisfied by our model are presented.

For the cold-start phase, we evaluate parameters α , β, λ and δ which represent power (effect) of the popularity score of the event, power of the quality score of the event, power of the distance factor and power of the time factor, respectively. We tested different values for these parameters and studied 136 events' scores and their probability of being recommended to 20 random users. Suggested values for these parameters $\alpha = 1$, $\beta = 2$, $\lambda = 1$ and $\delta = 5$ in Chapter 3 showed the best results for the sample events' scores. However, these parameters need to be tuned over the time upon more events.

In the content-based phase, in the optimal IDF score for informative words' computation, a coefficient of 0.03 is used. This coefficient expresses that a word is ideally informative if it appears in 3% of all events descriptions. Moreover, the linear combination coefficient $\eta = 0.5$ and coefficient $\mu = 1.1$ are employed in content-based computations. These values have been tested on the 136 sample events' descriptions. All words with their frequencies have been drawn and studied. We observed that informative and important words appeared in 3% of all descriptions. The coefficients $\eta = 0.5$ and $\mu = 1.1$ showed the best results for the 136 sample events' scores.

To continue the directions of study in this line of research, there are some natural extensions of the problem that we can mention. Another phase that can be considered for our model is *social network-based recommendation* which users' information and social connections on social networks are employed to recommend events which are

popular among their friends on these social networks. The additional data about user and their friends on social networks such as Facebook and Instagram can be used to improve the prediction accuracy of a recommender system. At this point, users can login with their Facebook account to the Brüha website. Moreover, if Brüha could update its application and website, some implied ratings can be learned from user's activities. For example, if the spent time for reading an event's information by a user is computed, it can be used as an implied rating of the user. In another extension, the evaluation of our proposed model can be done on a larger number of events and users. To perform the mentioned evaluation approaches in Chapter 2, a huge number of users and events are required in the system. Since Brüha is a newly established platform, the number of users, particularly active users, and events are insufficient to be able employ these evaluation metrics and measures. However, in the future, when more users and events are added to the system over time, these metrics are applicable.

Bibliography

[AT05] Gediminas Adomavicius and Alexander Tuzhilin. Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions. *IEEE transactions on knowledge and data engineering*, 17(6):734–749, 2005.

[BC92] Nicholas J Belkin and W Bruce Croft. Information filtering and information retrieval: Two sides of the same coin? *Communications of the ACM*, 35(12):29–38, 1992.

[BHK98a] John S Breese, David Heckerman, and Carl Kadie. Empirical analysis of predictive algorithms for collaborative filtering. In *Proceedings of the Fourteenth conference on Uncertainty in artificial intelligence*, pages 43–52. Morgan Kaufmann Publishers Inc., 1998.

[BHK98b] John S Breese, David Heckerman, and Carl Kadie. Empirical analysis of predictive algorithms for collaborative filtering. In *Proceedings of the Fourteenth conference on Uncertainty in artificial intelligence*, pages 43–52. Morgan Kaufmann Publishers Inc., 1998.

[Bur02] Robin Burke. Hybrid recommender systems: Survey and experiments. *User modeling and user-adapted interaction*, 12(4):331–370, 2002.

[Ged13] Fatih Gedikli. *Recommender systems and the social web: Leveraging tag-*

ging data for recommender systems. Springer Science & Business Media, 2013.

[JZFF10a] Dietmar Jannach, Markus Zanker, Alexander Felfernig, and Gerhard Friedrich. *Recommender systems: an introduction*. Cambridge University Press, 2010.

[JZFF10b] Dietmar Jannach, Markus Zanker, Alexander Felfernig, and Gerhard Friedrich. *Recommender systems: an introduction*. Cambridge University Press, 2010.

[MI04] Yutaka Matsuo and Mitsuru Ishizuka. Keyword extraction from a single document using word co-occurrence statistical information. *International Journal on Artificial Intelligence Tools*, 13(01):157–169, 2004.

[PHG+00] David M Pennock, Eric Horvitz, C Lee Giles, et al. Social choice theory and recommender systems: Analysis of the axiomatic foundations of collaborative filtering. In *AAAI/IAAI*, pages 729–734, 2000.

[RRS11] Francesco Ricci, Lior Rokach, and Bracha Shapira. *Introduction to recommender systems handbook*. Springer, 2011.

[RV97] Paul Resnick and Hal R Varian. Recommender systems. *Communications of the ACM*, 40(3):56–58, 1997.

[TTK+12] Mika Timonen, Timo Toivanen, Melissa Kasari, Yue Teng, Chao Cheng, and Liang He. Keyword extraction from short documents using three levels of word evaluation. In *International Joint Conference on Knowledge Discovery, Knowledge Engineering, and Knowledge Management*, pages 130–146. Springer, 2012.

[WCN12] Yuanyuan Wang, Stephen Chi-Fai Chan, and Grace Ngai. Applicability of demographic recommender system to tourist attractions: a case study on trip advisor. In *Proceedings of the The 2012 IEEE/WIC/ACM International Joint Conferences on Web Intelligence and Intelligent Agent Technology-Volume 03*, pages 97–101. IEEE Computer Society, 2012.

[WX08] Xiaojun Wan and Jianguo Xiao. Collabrank: towards a collaborative approach to single-document keyphrase extraction. In *Proceedings of the 22nd International Conference on Computational Linguistics-Volume 1*, pages 969–976. Association for Computational Linguistics, 2008.

Appendix A

Content-based Pseudocode

In this section, the pseudocode of our content-based filtering phase is presented.

Variables and constants:

D_t: event descriptions containing word t

p: number of users

m: number of all event descriptions

r: number of categories

$E[1,,m]$: an array of m events

$E[i]$.title: the title of the i-th event

$E[i]$.description: the description of the i-th event

$K[1,...,k]$: an array of k keywords

$C[1,\cdots,r]$: an array of r categories

$C[i]$.descriptions: event descriptions in category i

$f_i[]$: an array of fragments for description i

The following constants are used to tune the model and they can be changed over

the time:

$n_0 = 0.03 \times m$

$\mu = 1.3$

$\eta = 0.5$: a weight used for giving more emphasis to either corpus or category score.

Algorithm 1 - preprocessing

Input: $E[1, \cdots, m]$, an array of m events

Output: $E[1, \cdots, m]$, an array of m events' descriptions which have been preprocessed and an array $f[]$ that each element of it are each event's description fragments

For each $i \in [1, \cdots, m]$ **do**:

$E[i]$.description $\leftarrow$ add $E[i]$.title to $E[i]$.description

$E[i]$.description $\leftarrow$ remove all stop words and names from $E[i]$.description

$f_i[] \leftarrow$ break description by question mark, comma, semicolon, "and", "but", "or"

\# $f_i[]$ is an array that each element of it are fragments of description i

return $E[].description$, $f[]$

Algorithm 2 - computing corpus level score of keywords

Input: $E[].descriptions$, all events' descriptions

Output: A vector $scoreCorpus = (s_1, \cdots, s_k)$, where each element of it represents each keyword's score in corpus level

For each $j \in [1, \cdots, k]$ **do**:

$D_j \leftarrow$ the number of $E[].descriptions$ containing word j

IDF(j) $\leftarrow log(\frac{m}{D_j})$

\# IDF score for each word j

compute $K[i].docFrequency$

\# $K[i].docFrequency$ is the frequency of keyword i in $E[]$

$FW(j) \leftarrow \mu \times |log \frac{K[i].docFrequency}{n_0}|$

\# $FW(j)$, frequency weight score for each word j is the assumed IDF score for the word j to be optimally informative

$S_{corpus}[j] \leftarrow IDF(j) - FW(j)$

\# corpus level score of each word j

$scoreCorpus[j] \leftarrow \frac{S_{corpus}[j]}{\max_{l \in \{1, \cdots, k\}} S_{corpus}[l]}$

\# we divide each $scoreCorpus[j]$ by the maximum value to be between [0,1]

return scoreCorpus[];

Algorithm 3 - computing category level score of keywords

Input: $E[].descriptions$, all events' descriptions, and $f[]$ fragments of each description

Output: An array $scoreCategory[][]$ which each entry $category[j][s]$ of it represents category score of word j in the category s

For each $t \in [1, \cdots, r]$ **do**:

For each $j \in [1, \cdots, k]$ **do**:

compute $count.f_j$

\# $count.f_j$ the number of fragments which contain word j

compute $length.f_i[]$

\# $length.f[i]$ is the length of the fragment i

$ifl(j) \leftarrow \frac{1}{(\frac{1}{count.f_j}) \times \sum_{j \in f_i[]} length.f[i]}$

\# $ifl(j)$, inverse average fragment length of word j, is the average number of words in the set of fragments the word appears in.

compute $count.c_j$

\# $count.c_j$ is the number of categories which contain word j

$ic(j) \leftarrow \frac{1}{count.c_j}$

\# $ic(j)$ is the inverse count of categories for the word j

$RCoR(j) \leftarrow ifl(j) + ic(j)$

\# $RCoR(j)$ is the relevance of the word j in the corpus

$P_{c,d}[j][t] \leftarrow \frac{|D_j \cap C[t].descriptions|}{|D_j|}$

\# $P_{c,d}[j][t]$: the probability that an event description D containing word j occurs in the category t

$P_{d,c}[j][t] \leftarrow \frac{|D_j \cap C[t].descriptions|}{|C[t].descriptions|}$

$P_{d,c}[j][t]$: the probability that for a given description D in the category t contains word j

$RCaT[j][t] \leftarrow (P_{c,d}[j][t] + P_{d,c}[j][t])$

$RCaT[j][t]$ is the relevance of the word j in the category t

$scoreCategory[j][t] \leftarrow RCoR[j] \times RCaT[j][t]$

return $scoreCategory[][]$

Algorithm 4 - merging corpus and category scores

Input: corpus score and category score of each keyword

Output: the array $Score = (s_1, \cdots, s_r)$, where each element of it represents each categorie's score its keyword's using its corpus and category scores

For each $t \in [1, \cdots, r]$ **do**:

For each $j \in [1, \cdots, k]$ **do**:

final.score[j][t] $\leftarrow \eta \times$ corpus[j] + (1-η)$\times$ category[j][t]

corpus[j] and category[j][t] are obtained by algorithm 1 and 2 respectively

For each $t \in [1, \cdots, r]$ **do**:

$Score[t] \leftarrow \sum_{\forall j} Score[j][t]$

return Score[];

Algorithm 5 - measure the similarity between user and event descriptor vectors

Input: all users and events descriptor vectors

Output: similarity score between any two user and event descriptor vectors

For each $i \in [1, \cdots, p]$ **do**:

For each $j \in [1, \cdots, m]$ **do**:

similarity[i][j] $\leftarrow \frac{u_i.e_j}{|u_i|.|e_j|}$

each u_i and e_j are vectors of length r, the number of categories, which each element of them represents the score of corresponding category for u_i and e_j.

return similarity[][];

similarity[][] is the similarity matrix where the rows are users and columns are events and each entry similarity[i][j] represents the similarity between two descriptor vectors for user i and event j

Appendix B

About Brüha

Brüha established in Hamilton ON, Showdom Inc. was formed in May 2012 and has since merged with Brüha Inc. as of December 2015. Showdom Inc operating as Brüha, is a local entertainment and online ticket provider. It is a website, iOS and Android product that streamlines the process of finding local things to do and buy/sell event tickets. Brüha has developed software products including a website, an iOS application and an Android application. In its simplest form customers advertise events and sell tickets on Brüha, where users come to the products to explore events happening in their area, filter for content they are interested in and purchase tickets as required. In the following, some screen shots of Brüha's website and iOS application are brought.

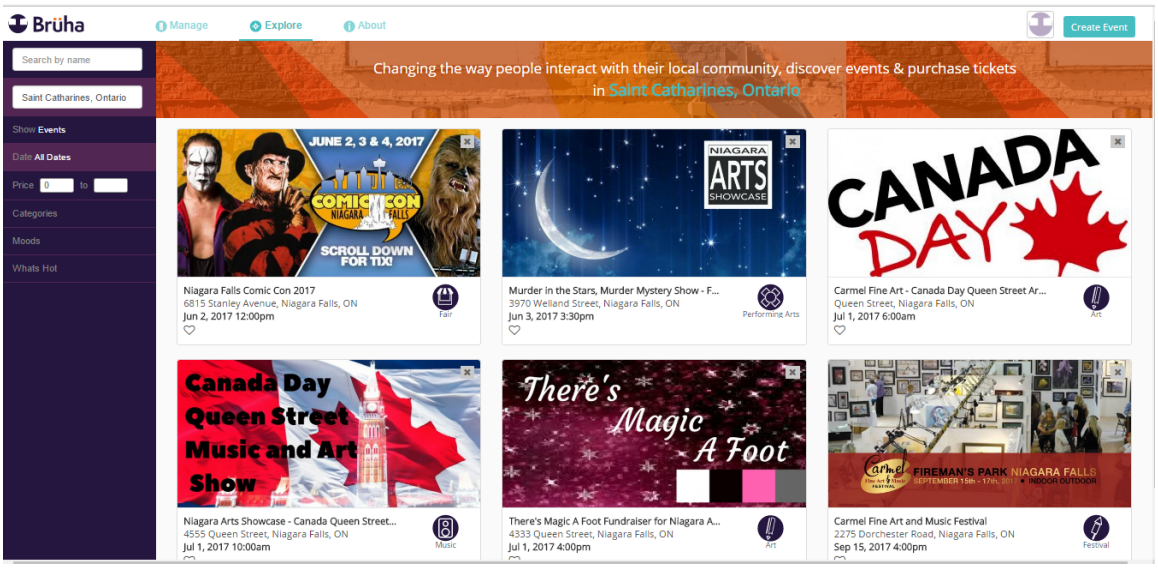

Figure B.1: Brüha's website

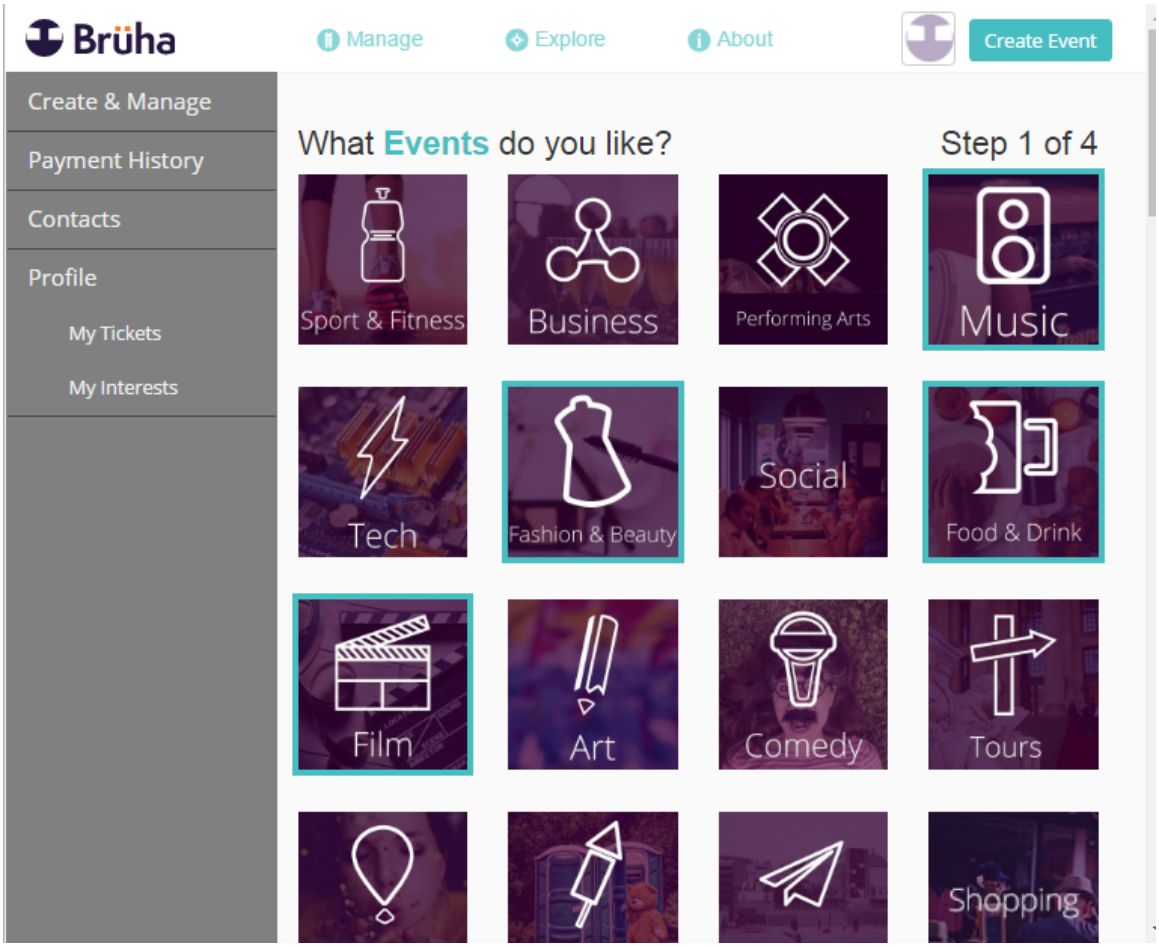

Figure B.2: User's interests on Brüha's website

Figure B.3: An event's description on Brüha's website

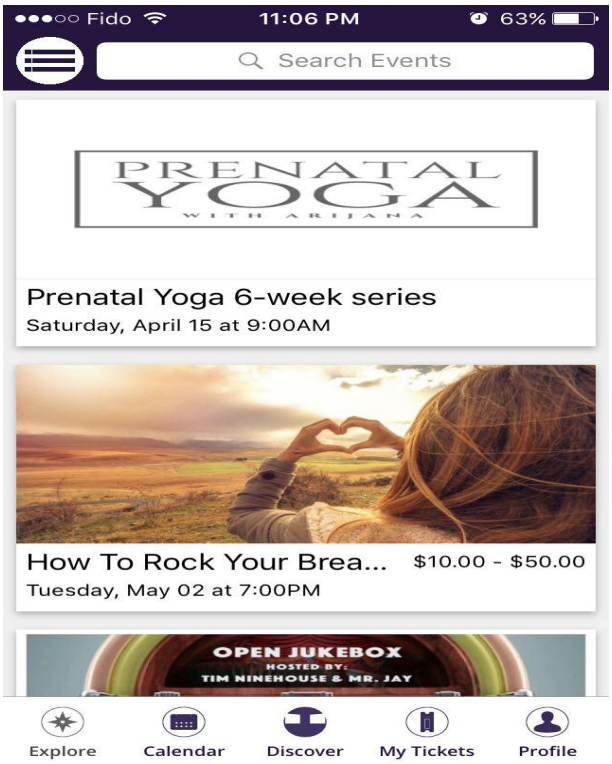

Figure B.4: Brüha's iOS application

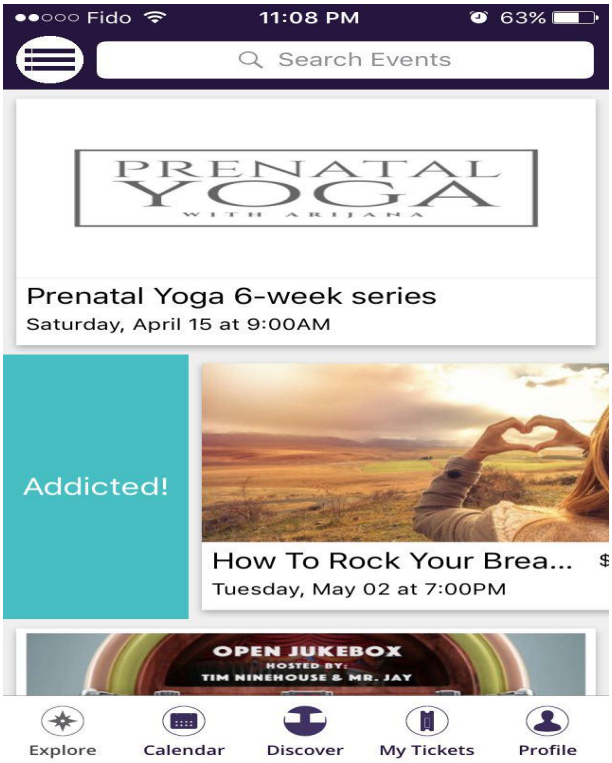

Figure B.5: Brüha's iOS application

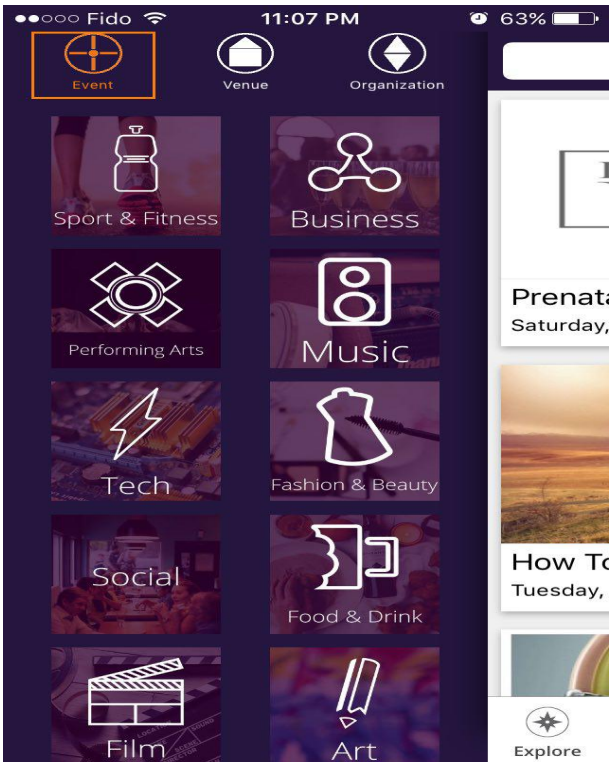

Figure B.6: Brüha's iOS application

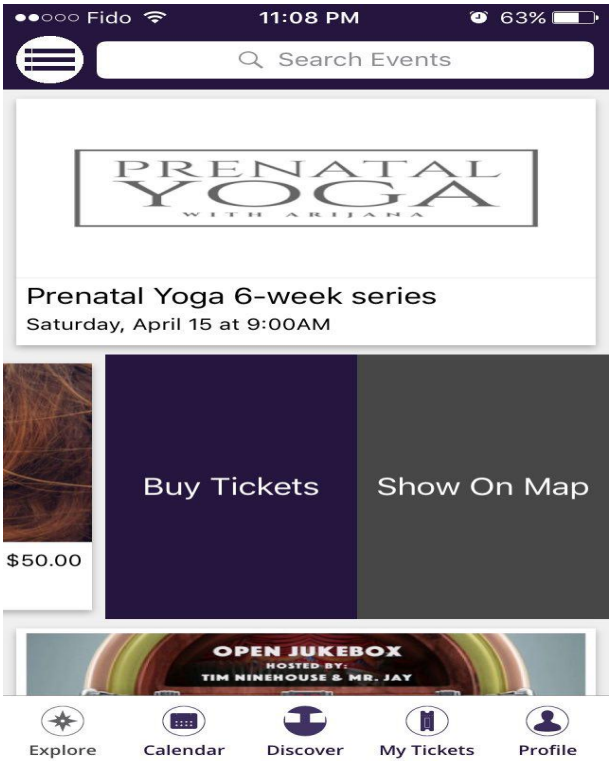

Figure B.7: Brüha's iOS application

Appendix C

Complete tables for parameters a and b

Here, complete tables for parameters a and b which are the representation for distance and time difference in LTPO recommendation are brought.

$$b = \begin{cases} 11, & 0\text{s} < t \leq 86400\text{s} \\ 10, & 86400\text{s} < t \leq 86400\text{s} \times 2 = 172800s \\ 9, & 86400\text{s} \times 2 < t \leq 86400\text{s} \times 3 = 259200s \\ 8, & 86400\text{s} \times 3 < t \leq 86400s \times 4 = 345600\text{s} \\ 7, & 86400\text{s} \times 4 < t \leq 86400s \times 5 = 43200\text{s} \\ 6, & 86400\text{s} \times 5 < t \leq 86400s \times 6 = 518400\text{s} \\ 5, & 86400\text{s} \times 6 < t \leq 86400s \times 7 = 604800\text{s} \\ 4, & 86400\text{s} \times 7 < t \leq 86400s \times 14 = 1209600\text{s} \\ 3, & 86400\text{s} \times 14 < t \leq 86400s \times 21 = 1814400\text{s} \\ 2, & 86400\text{s} \times 21 < t \leq 86400s \times 28 = 50803200\text{s} \\ 1, & t > 86400s \times 28 = 50803200\text{s} \end{cases}$$

$$a = \begin{cases} 18, & 0km < d \leq 1km \\ 17, & 1km < d \leq 2km \\ 16, & 2km < d \leq 3km \\ 15, & 3km < d \leq 4km \\ 14, & 4km < d \leq 5km \\ 13, & 5km < d \leq 6km \\ 12, & 6km < d \leq 7km \\ 11, & 7km < d \leq 8km \\ 10, & 8km < d \leq 9km \\ 9, & 9km < d \leq 10km \\ 8, & 10km < d \leq 12km \\ 7, & 12km < d \leq 14km \\ 6, & 14km < d \leq 16km \\ 5, & 16km < d \leq 18km \\ 4, & 18km < d \leq 20km \\ 3, & 20km < d \leq 40km \\ 2, & 40km < d \leq 60km \\ 1, & d > 60km \end{cases}$$

Appendix D

Defined Categories and Keywords

Here, our defined categories and most likely keywords associated with these categories are brought.

Film	Art	Performing Arts	Drink/Food	Outdoor	Sport & Fitness
Cinema	Painting	Comedy	Beer	Promenade	Flyboarding
Film Club	Gallery	Theatre	BREWERY!	Tours	Soccer
	Crawl	Comic	Club	Park	Throwing
	calligraphy	Amphitheatre	cider	Beach	Tournament
	Expo	Comedy Club	LIQUOR	Camp	Archery
	Artist	Arena	Pub	Adventure	Badminton
	exhibitor	Fairground	Bar	Sunny	Boat
	exhibition	Community Hall	Bourbon	Hike	racing
		Revue	Cafe	LAKE	Play
		Musical	Cheese	Walk	Racquet
		clown	CHERRY	mountain	Arena
			Dinner	Recreation	Sport(org)
			Ice Cream	Expo	Recreation
			BBQ	vehicle	Baseball
			Cheese	automobile	Boxing
			Diner		
			beverages		
			Nightclub		
			fun		
			pizza		
			Hard Liquor		
			dairy		
			chef		
			Recreation		
			healthy		
			Oyster		

Figure D.1: 13 defined categories and corresponding keywords

Music and Party	Festival	Science/Technology
Music	Car show	Conference
Dance	Festival	Science
Party	Carnival	Workshops
Solo	Celebration	School Event
swing	Show	School
Band		Amphitheatre
Celebration		technology
DJ		Academic
Waterworks		Community Hall
Club		Biology
jazz		
cello		
albume		
pop		
song		
riff		
musical		
bass guitar		
rhythm guitar		
drums		
vocal		
Heavy metal		
fun		
hip hop		
Dancehall		
Recreation		
Rock and roll		
Bluegrass		

Play & social	Fashion & Beauty	religion	Kids friendly
Play	Fashion	Spiritual	face painting
Trivia	collection	Church	jumpy castles
Casino	Dior	religion	bouncy castle
Tournament	Chanel		
games			
fun			
Toastmaster			
social skill			
social			
game			
leadership			
Sociology			

Figure D.3: 13 defined categories and corresponding keywords